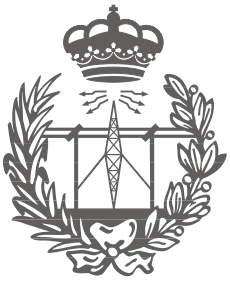
Escuela Técnica Superior de Ingeniería Informática

Ingeniería de la Salud mención en Bioinformática



Herramienta para el etiquetado de objetos en secuencias de video.

Object labelling tool for video sequences.

Realizado por:

Jose Rodríguez Maldonado

Tutorizado por :

Rafael Marcos Luque Baena, Miguel Ángel Molina Cabello

Departamento:

Lenguajes y ciencias de la computación LCC

UNIVERSIDAD DE MÁLAGA

MALAGA, 12/2019

Fecha defensa:

Fdo.: El Secretario del tribunal

Herramienta para el etiquetado de objetos en secuencias de video.

Resumen:

Uno de los tres pilares sobre los que se sustenta la inteligencia artificial es la disponibilidad de datos públicos etiquetados. En el campo de la visión por ordenador puede llegar a resultar muy complicado, y sobre todo costoso, conseguir datos fiables y etiquetados, llegando a consumir casi la totalidad de recursos y tiempo en un proyecto. Revisando el estado del arte de las tecnologías disponibles para el etiquetado de datos distinguimos dos vertientes: enfoques automáticos, que no consiguen generar datos fiables, o enfoques manuales, centrados en buscar la perfección en el etiquetado que resultan muy costosos.

En este proyecto, se ha desarrollado una herramienta para generar conjuntos de datos etiquetados a partir de secuencias de videos que permitirá etiquetar datos de una forma bastante fiable y rápida.

Para poder etiquetar los datos de una forma semi-automática seguimos un procedimiento de 3 fases. En primer lugar el programa detecta los objetos que aparecen en las imágenes y extrae las trayectorias de los objetos en movimiento. A continuación, etiquetamos a mano las diferentes trayectorias. Finalmente, el programa extraerá para cada trayectoria tanto su etiqueta como todas las apariciones del objeto en los diferentes fotogramas del vídeo.

Con el filtro de Kalman detectamos las trayectorias del los objetos a lo largo del vídeo, que a su vez son detectados en la imagen mediante una segmentación que está basada en mapas auto-organizados. Envolveremos nuestro desarrollo bajo una interfaz pensada para que sea fácil de utilizar. El código de la aplicación ha sido implementado en Matlab, que es un lenguaje que destaca por su versatilidad y rapidez al trabajar con imágenes. La interfaz se basa también en Matlab, concretamente usa la librería Matlab appdesigner para el diseño de interfaces.

Finalmente entrenamos una red Faster R-CNN, que se usa en la detección de objetos, para comprobar los resultados para los diferentes conjuntos de datos generados. Y vemos, como se esperaba, que conforme aumentamos el nivel de automatización, añadimos más ruido. Podemos concluir que este enfoque semi-automático para el etiquetado de datos en secuencias de video supone un aporte muy interesante sobre el estado del arte actual al permitir generar conjuntos de datos etiquetados de imágenes de una forma rápida.

Palabras clave: video, extraer, imagenes, trayectoria, filtro de Kalman, Aprendizaje profundo, Mapa auto-organizado, etiquetado, generar datos

Object labelling tool for video sequences.

Abstract:

The availability of public labeled data is one of the three pillars for encouraging the evolution of artificial intelligence. Computer vision is one of the more expensive fields to acquire reliably labeled data. It requires a significant investment in both time and money. It is very common to spend about 90 % of a computer vision project just creating reliable labeled data. If we look over the state of the art techniques that are used to label data, we distinguish 2 choices: Automatic approaches, that cannot generate reliable data, and manual approaches that pursue labeling perfection. They offer plenty of options to define complex bounding boxes that still, are too expensive.

We have developed a semi-automatic tool that creates from video sequences labeled data which is reliable and much faster than any other manual approach.

The process used to label data semi-automatically consists of 3 different steps. First of all, the program detects the different objects that appear on the video frames, iteratively the program will match every new appearance of objects to its corresponding trajectory. Meanwhile, we will label the different object trajectories.

Once we are finished, the program will extract all the appearances of the object in the video with its corresponding label, creating datasets.

We are using the Kalman filter to track the trajectories of the different objects. We use a segmentation method, based on Self-organizing maps, to detect objects from a video frame. The development will be wrapped into an interface designed to be user-friendly. This tool has been developed using Matlab because it is a very dynamic and powerful language to work with images. The interface was developed using Matlab appdesigner library.

We will check the validity of the resulting datasets comparing the variance we get when training a Faster R-CNN with the different generated datasets. As we expected, we see that the bias we get increases as we increase the automation generating data. In conclusion, this tool stands as a new interesting approach that improves the current state of the art allowing us to get labeled data fastly, encouraging the evolution of computer vision.

Key words: video, extract, images, trajectories, Kalman filter, Deep Learning, Self-organizing map, labeling, generate data

Málaga, 21 de noviembre de 2019

Índice

I Introducción **1**

Introducción y visión general **3**

Motivación 3

Objetivo central 7

Estado del arte 7

Metodología y directrices seguidas 10

Tecnologías a utilizar 11

II Métodos **13**

1 Métodos **15**

1.1 Fundamentos del procesamiento de imágenes 17

1.1.1 Imagen digital 17

1.1.2 Procesamiento de imágenes 17

1.2 Redes neuronales artificiales 20

1.2.1 Mapas auto-organizados 21

1.2.2 Aprendizaje profundo y redes convolucionales 23

1.3 Algoritmo de Kalman 24

1.4 Principios metodológicos 24

III Desarrollo **27**

2 Desarrollo y análisis del programa **29**

2.1 Detección de objetos 31

2.2 Extracción de trayectorias 33

2.3 Desarrollo interfaz 36

2.3.1 Paso 1: Definir etiquetas 37

2.3.2 Paso 2: Etiquetado de trayectorias 39

2.3.3 Paso 3: Métricas y almacenamiento de extracción 46

Conclusiones y líneas futuras **51**

Bibliografía **63**

Índice alfabético **65**

Índice de figuras

1 Diagrama Gantt del proyecto 11

1.1 Resumen aplicación 16
1.2 Imagen obtenida de Wikipedia [Wikipedia, 7 30] sobre la corrección gamma 18
1.3 Caso real: Segmentación. 20
1.4 Descripción de red neuronal artificial. 21
1.5 Imagen extraída de Wikipedia [MartinThoma, 2016]: Ilustración de SOM. 22
1.6 Ilustración del algoritmo de Kalman para una trayectoria. 24
1.7 Ilustración del método científico. 25

2.1 Segmentación FSOM 32
2.2 Ejemplo segmentación y extracción de recuadros 33
2.3 Diagrama algoritmo de Kalman 35
2.4 Elegir carpetas 38
2.5 Añadir etiqueta y borrar etiqueta 39
2.6 Ejemplo del paso 2 de la interfaz 40
2.7 Etiquetado: Paso 1 (Puntos suspensivos indican que son acciones sobre ventanas emergentes.) 42
2.8 Etiquetado: Paso 2 43

2.9 Detección de trayectorias en renacuajos 44
2.10 Detección de trayectorias en pez cebra 44
2.11 Detección de trayectorias en hormigas 45
2.12 Detección de trayectorias en vehículos 45
2.13 Ejemplo del paso 3 de la interfaz 47
2.14 Estructura de carpetas generada. 48
2.15 Ejemplo algoritmo de adelgazamiento. 52

Parte I

Introducción

Introducción y visión general

Contenido

Motivación **3**

Objetivo central **7**

Estado del arte **7**

Metodología y directrices seguidas **10**

Tecnologías a utilizar **11**

Motivación

En estos últimos años la **inteligencia artificial (IA)** ha pasado de ser ciencia ficción a afectar directamente la forma en la que vivimos, y la forma en la que trabajamos. Expertos en IA como Andrew Ng afirman que tendrá el mismo impacto en la industria que el que tuvo la electricidad[Drummond, 2017].

Tan descomunal ha sido el crecimiento de la IA durante esta última década, que algunos estudios apuntan que impulsará el producto interior bruto (PIB) global en un 14 % para 2030, lo que se traduce en un impacto económico sobre el PIB global de **15,7 billones de dólares** [PwC, 2017]. Dentro de la inteligencia artificial destaca el sector de la visión por ordenador. Según un reciente estudio de O'Reilly

[Naimat, 2016] sobre el mercado de la IA, el reconocimiento de imágenes es el segundo área donde más capital se está invirtiendo en Estados Unidos.

La **visión por ordenador** consiste en un conjunto de herramientas y métodos que permiten obtener, procesar y analizar imágenes del mundo real a través de un ordenador[INFAIMON, 1 18]. Pese a que el interés hacia esta rama de la IA se ha despertado recientemente (concretamente a partir de 2012 cuando AlexNet ganó el ImageNet Challenge), esta disciplina lleva investigándose mas de 60 años [Demush, 2019].

Uno de los **artículos mas influyentes** publicados en torno a la visión por ordenador es obra de dos neurofisiólogos (David Hubel y Torsten Wiesel) en 1950 [Hubel and Wiesel, 1959]. Concluyeron que en el cortex visual primario tenemos neuronas tanto simples como complejas, y que el procesamiento visual empieza detectando estructuras sencillas, como la orientación de los bordes [HUBEL, 1981]. Podemos notar que las conclusiones obtenidas describen los principios en los que se basa el aprendizaje profundo.

Desde la publicación de este artículo tenemos varios **eventos** muy importantes:

- **1959**: La aparición del primer escaner digital [Lewis, 2013]
- **1963**: "Machine percepton of three-dimensional solids"[Roberts, 1963].El objetivo de esta tesis era extraer información tridimensional de cuerpos geométricos a partir de su representación en dos dimensiones.
- **1982**: Kunihiko Fukushima construyó una red neuronal auto-organizada formada por neuronas sencillas y complejas, tal y como David Hubel and Torsten Wiesel describian, que pudieran reconocer patrones sin importar los cambios de posición [Fukushima, 1982].
- **1989**: Yann LeCun aplicó retropropragación (backpropagation) a la red convolucional de Fukushima, para reconocer caracteres. El trabajo de Yann LeCun

dió lugar a la red LeNet-5 y al famoso conjunto de datos MNIST [LeCun, 1989].

- **1990**: Cambia el rumbo de la visión por ordenador como se había estudiado hasta el momento; dejó de centrarse en la creación de modelos 3D y empezaron a aparecer más trabajos relacionados con el el reconocimiento de objetos [Demush, 2019]. Una muestra de ello es el trabajo de David Lowe [Lowe, 1999].

- **2001**: Surje el *Viola/Jones face detector* [Viola and Jones, 2001], el primer detector de caras, este no usaba aprendizaje profundo sino Adaboost, aun hoy día se sigue usando.

- **2006 - 2012**: Se inicia la competición anual Pascal VOC en el que se ofrecía un conjunto de datos estándar para clasificación y que permitía reconocer que modelos funcionaban mejor para el reconocimiento de clases [PASCAL2, 2019].

- **2012**: Irrumpen las redes convolucionales mejorando el error en las predicciones de un 26 % a un 16.4 %. Desde este año, el ganador del concurso siempre han sido redes convolucionales [ImageNet, 2012].

Las **redes convolucionales**, así como algoritmos de aprendizaje automático en general, han ganado tanta **fama** recientemente, pese a que existen desde los años 80, por **tres motivos** principales [Juan de Antonio, 2018]:

- Mejora en la **capacidad computacional**. La capacidad de computación de la que se dispone hoy día es mucho mayor que la que había en los años 90, lo que nos permite utilizar modelos mucho mas complejos y pesados.

- **Mejora de los algoritmos**.

- **Mayor disponibilidad de datos**

El **«boom»** de la visión por ordenador esta permitiendo desarrollar proyectos muy futuristas en múltiples sectores. Por ejemplo, dentro del sector de la **automoción**, resulta llamativo el caso de la empresa TuSimple [AWS, 2019]. TuSimple es una empresa de trasporte que a día de hoy ha alcanzado con sus camiones un nivel de autonomía 4. Un nivel 4 de autonomía consiste en que siempre debe ir un operario dentro de la cabina del camión, para poder intervenir si fuera necesario; se espera que para 2020 ya sean plenamente automáticos.

A la hora de afrontar un **proyecto de visión por ordenador**, al igual que cualquier otro proyecto de IA, solemos encontrarnos con las siguientes **fases** [Heras, 2018]: Planificación del proyecto, Recopilación y etiquetado de datos, Búsqueda de modelos, Mejora de modelo, Evaluación, Puesta en producción y Mantenimiento. La peculiaridad de los proyectos de visión por ordenador es que la fase de recopilación de datos y evaluación suele consumir muchos de los recursos del proyecto [Nakhuda, 2019].

En el gran **salón de la fama de la IA** es importante dejar un espacio para todos aquellas personas dedicadas a etiquetar conjuntos de datos. Quiero destacar el proyecto de ImageNet [Brownlee, 2019], una base de datos de imágenes en las que se etiquetaron 14 millones de imágenes en un plazo de 9 años de forma completamente manual. Resulta fácil imaginar el impacto que causaría encontrar un **enfoque semi-automático** que permita clasificar objetos de una forma mucho mas rápida y fiable. Esta es la motivación de la que surge mi trabajo de fin de grado (TFG).

El **impacto** más claro y directo sería la reducción del coste tanto económico como en tiempo para un proyecto de IA. Al reducir costes y tiempo parece lógico imaginar que tendríamos un aumento de proyectos de IA, al volverse mas asequibles y rápidos; algo que afectaría de forma transversal a todas y cada uno de los sectores en los que está involucrados la visión por ordenador como: automoción, salud, asistencia, et al.

Objetivo central

El objetivo principal de este proyecto consiste en el estudio e implementación de una alternativa a la detección manual y automática de objetos, un enfoque semi-automatico basado en la detección de las trayectorias de objetos en secuencias de vídeo. Desde el punto de vista económico, el objetivo consiste en reducir los costes a la hora de afrontar un etiquetado de imágenes. Tecnológicamente, esta aplicación ayudará a generar un conjuntos de datos etiquetados de forma fiable y rápida, además, deberá facilitar la interacción del usuario con la herramienta, y generar conjuntos de datos de forma que podamos utilizarlos directamente para entrenar un modelo.

Este enfoque esta constituido por la consecución de los siguientes puntos:

- El estudio de los distintos enfoques y técnicas actuales en lo relativo a la inteligencia artificial, concretamente en el campo de la visión por ordenador para el etiquetado de datos.
- Detección de objetos a través de la interfaz
- Extracción y almacenamiento de trayectorias
- Desarrollo de interfaz

Estado del arte

Es muy importante planificar bien la forma en la que vamos a etiquetar los datos para un proyecto de visión por ordenador, de lo contrario podríamos estar generando costes tanto monetarios [INGEDATA, 2019a] como en tiempo [INGEDATA, 2019b].

Para cualquier proyecto que requiera datos etiquetados fiables podríamos contratar a alguna empresa de *Outsourcing* especializadas. Estas empresas dan buenos

resultados pero suelen tener algunos problemas como altos costes, o pérdida de calidad del conjunto de datos [RAINERI, 2019]. Otra alternativa interesante a la hora de externalizar el etiquetado es el Crowdsourcing, permite distribuir el trabajo del etiquetado en tantos autónomos como sea necesario además de bajar los costes [Floren, 2012]. El Crowdsourcing es parecido al Outsourcing pero el equipo está formado por una cooperación entre diferentes autónomos, además suele ser muy rápido y ahorra muchos costes, el problema es que puede comprometer la calidad de los datos [AltexSoft, 2018]. En algunos casos se llega a hacer gala del famoso refrán "si quieres algo bien hecho hazlo tú mismo" y se opta por hacer un etiquetado de las imágenes a mano de forma interna, como mucho contratando a algún autónomo para que ayude temporalmente; proceso que requiere de mucho tiempo [AltexSoft, 2018].

Antes de decantarnos por etiquetar imágenes de forma manual e interna, podemos plantearnos algunas alternativas como generar datos sintéticamente [Surma, 2019]. Una alternativa interesante sería diseñar algún programa que sea capaz de etiquetar los datos automáticamente. El mayor problema que tienen los programas encargados de etiquetar automáticamente imágenes es que necesitan de algoritmos de aprendizaje supervisado. Estos algoritmos suelen estar entrenados normalmente en entornos poco ruidosos, ademas, están entrenados para reconocer solo un conjunto pequeño de clases [Xin-Jing Wang and Ma, 2006].

La mayoría de las soluciones están orientadas a hacer el etiquetado de una forma mas fácil. Por ejemplo, el proyecto FastAnnotationTool [Axa, 2019] consiste de una pequeña interfaz que nos va a permitir establecer cajas que podemos etiquetar en una forma relativamente sencilla.

De entre los proyectos más usados hoy día para el etiquetado de imágenes podemos distinguir 2 vertientes. Por un lado tenemos proyectos más centrados en proporcionar facilidades en el etiquetado; por otro lado tenemos los proyectos que

buscan la perfección en el etiquetado, capaces de etiquetar figuras muy complejas. Algunos de los proyectos mas usados hoy día para el etiquetado de imágenes manual son:

- CVAT: Computer Vision Annotation Tool [opencv, 9 01]
- labelme: Image Polygonal Annotation with Python [wkentaro, 9 01]
- VIA: VGG Image Annotator [Abhishek Dutta and Zisserman, 9 01]
- Labelbox [Labelbox, 9 01]

De entre estos proyectos, me parece interesante una funcionalidad de CVAT citado anteriormente. Cuando trabaja con videos es capaz de interpolar la trayectoria de un objeto dada su posición en 2 fotogramas diferentes.

Encontramos propuestas muy similares en los proyectos de etiquetado semiaumático de imágenes. Centran sus esfuerzos en detectar los diferentes objetos que hay en una imagen junto con su etiqueta automáticamente. Un ejemplo de una aplicación de uso general de este estilo sería Anno-Mage [virajmavani, 2019].

Las limitaciones de esta aplicación son prácticamente las mismas que comentábamos anteriormente para los enfoques automáticos. A diferencia del enfoque automático podemos corregir a mano errores en el etiquetado automático, o recuadros demasiado holgados.

La herramienta que vamos a desarrollar se basa en algunos de los conceptos comentados anteriormente. En vez de diseñar un clasificador que sea capaz de etiquetar objetos, nos centraremos en diseñar un objeto que detecte posibles objetos dentro de una imagen.

A priori no parece un gran avance puesto que aunque detectamos los recuadros que engloban los objetos no sabemos que etiqueta tienen y tendríamos que etiquetar todos los objetos. Pero, vamos a servirnos de la temporalidad de los datos que

ofrece los vídeos y del algoritmo de Kalman, que explicaremos más en detalle a continuación, para detectar la trayectoria de los posibles objetos a lo largo del vídeo.

Para cada etiqueta podemos extraer todas la apariciones del objeto etiquetados en el vídeo por su trayectoria. Pudiendo controlar que objetos etiquetamo,s y solo tener que etiquetar cada objeto una vez, esperamos reducir sensiblemente el tiempo de etiquetado.

Metodología y directrices seguidas

El trabajo se desarrollará haciendo uso del **método científico** [Academy, 2019]. Por consiguiente, tendremos en cuenta la rigurosidad y reproducibilidad del experimento. Además, utilizaremos una metodología iterativa incremental [Caycho, 2019], en la que partimos de una base sobre la que añadiremos nuevas características sin perjudicar el trabajo ya desarrollado. De cara a la implementación, la metodología utilizada permitirá que el trabajo sea extensible, sencillo y se encuentre bien documentado.

El principal método en el cual se basa el trabajo explicado es un método de **segmentación basado en mapas auto-organizados** para la detección de objetos en primer plano (algoritmo FSOM [López-Rubio et al., 2011]). Sobre los objetos detectados se aplica el filtro de Kalman para realizar un seguimiento de los mismos y reconocer sus trayectorias.

Se deja a continuación un Diagrama de Gantt donde se indican las horas dedicadas a cada una de las diferentes fases del proyecto.

El vídeo que vamos a utilizar para testar la aplicación es sb-camera2-0750am-0805am. Secuencia de vídeo extraída del conjunto de datos Next Generation Simulation (NGSIM) [administration, 2019], que ofrece la administración federal de autovías (Federal Highway Administration, FHWA).

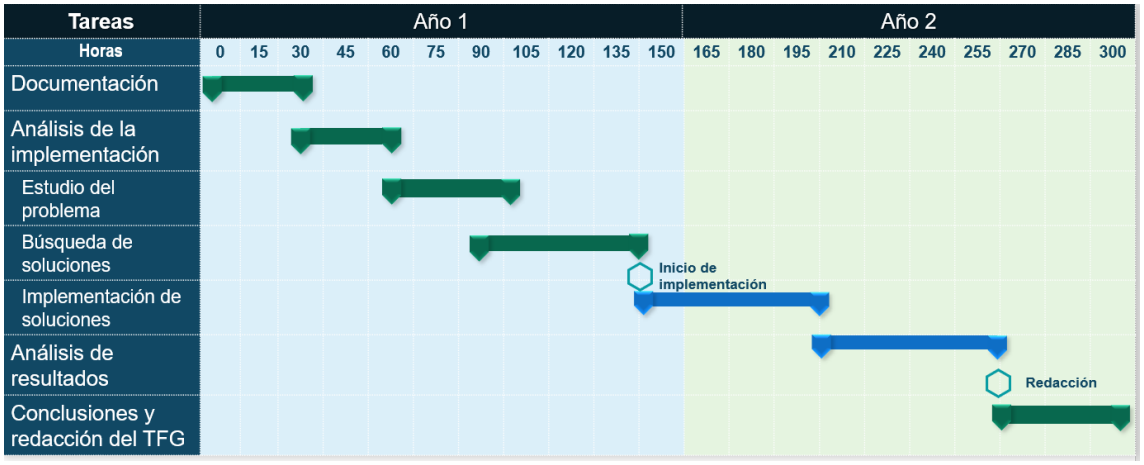


Figura 1: Diagrama Gantt del proyecto

Tecnologías a utilizar

Este proyecto será desarrollado utilizando Matlab. Matlab destaca de entre otros lenguajes por:

Entorno de desarrollo Ofrece un entorno de desarrollo de alta productividad, capaz de adaptarse a tareas específicas.[MATLAB, 2019]

Lenguaje Es un lenguaje fácil de aprender que dispone de gran cantidad de algoritmos de gran calidad optimizados en los que se puede confiar.[MATLAB, 2019]

Toolboxes y apps Matlab dispone de una gran cantidad de toolboxes para distintas aplicaciones, como por ejemplo para Machine Learning[MATLAB, 2019]

Rendimiento Matlab dispone de multithreading integrado, optimizando el cálculo en equipos multinúcleo. Llegando a ser entre 3 y 120 veces mas rápido que R. Además ofrece para tareas que requieran de una alta carga computacional la Parallel Computing Toolbox para ejecutar varios motores de Matlab. Matlab es un lenguaje que rinde especialmente bien trabajando con matrices, algo muy conveniente para el trabajo que vamos a acometer [MATLAB, 2019].

Despliegue fácil Matlab cuenta con todo un repertorio de herramientas para desarrolladores como depurador interactivo, programación orientada a objetos, analizados de rendimiento de código, marco de pruebas unitarias, integración de control de versiones y un desarrollador de interfaz de usuario. Además permite compartir el trabajo realizado mediante la publicación del codigo en múltiples formatos [MATLAB, 2019].

Documentación, soporte y comunidad Matlab no solo dispone de una documentación bastante exhaustiva y una grán comunidad de usuarios, también cuentan con un soporte profesional con mas de 200 expertos en todo el mundo con dedicación exclusiva a resolver preguntas y solucionar problemas [MATLAB, 2019].

Parte II

Métodos

Capítulo 1

Métodos

Contenido

1.1 Fundamentos del procesamiento de imágenes 17

1.1.1 Imagen digital 17

1.1.2 Procesamiento de imágenes 17

1.2 Redes neuronales artificiales 20

1.2.1 Mapas auto-organizados 21

1.2.2 Aprendizaje profundo y redes convolucionales 23

1.3 Algoritmo de Kalman 24

1.4 Principios metodológicos 24

Sinopsis

A continuación haremos un recorrido sobre los conocimientos teóricos necesarios para poder entender plenamente el trabajo realizado esclareciendo la visión global del proyecto.

Empezaremos reforzando conceptos sobre la imagen digital y procesamiento de imágenes. Ahondaremos en los conceptos de inteligencia artificial necesarios para entender el proceso de segmentación basado en mapas auto-organizados. Seguidamente, daremos una visión general del filtro de Kalman, que se usa para extraer las trayectorias de los objetos en movimiento. Y finalmente, comentaré los principios metodológicos seguidos a lo largo del proyecto.

La herramienta que se va a implementar será capaz de extraer detectar objetos dentro de una imagen, usando una segmentación basada en mapas auto-organizados, así como predecir su trayectoria, usando el algoritmo de Kalman. Esas trayectorias son las que posteriormente etiquetaremos y organizaremos.

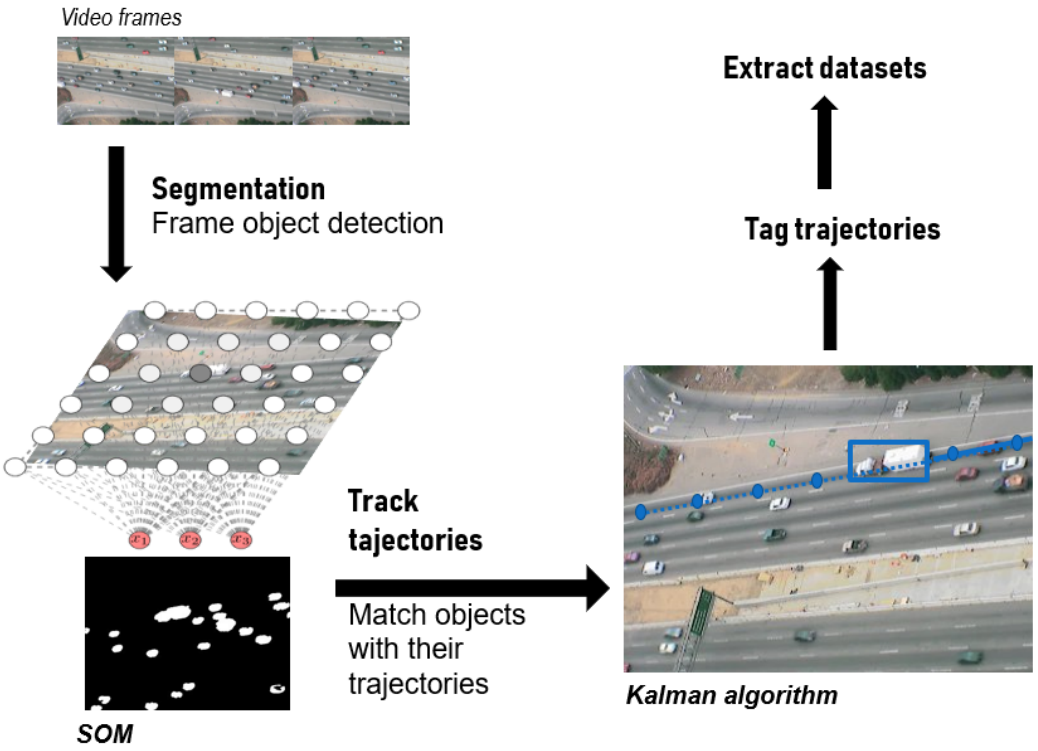


Figura 1.1: Resumen aplicación

1.1. Fundamentos del procesamiento de imágenes

1.1.1. Imagen digital

Las imágenes digitales se generan por una combinación de una fuente de "iluminación" (además de luz también puede ser infrarrojos, ultrasonidos, et. al) y la reflexión o absorción de energía de la escena que se esté capturando [def, 2005]. Este proceso transforma la energía entrante en un voltaje que dependerá de la entrada y el material del sensor, la respuesta de este sensor es digitalizada, lo que llamaremos píxeles, que en conjunto formarán la imagen [Osuna, 2019].

Una imagen se define por una función bidimensional, $f(x, y)$. Cuando x e y son finitos y discretos se habla de imagen digital. Una imagen digital está compuesta por un número finito de elementos llamados píxeles. x e y son coordenadas que nos permitirán localizar los diferentes píxeles dentro de una imagen y extraer su intensidad de color. A la intensidad de una imagen monocromática f en las coordenadas (x, y) se le denomina nivel de gris [fun, 2005].

1.1.2. Procesamiento de imágenes

Al digitalizar una imagen estamos convirtiendo el valor continuo de los píxeles a un equivalente digital mediante un muestreo de la imagen física [Soria, 2019]. Trabajar con imágenes digitales se convierte en un problema de trabajar con matrices.

Por tanto, procesar imágenes no es mas que aplicar transformaciones a estas matrices. Destacan dos ámbitos de aplicación [Pérez and Valente, 2018]:

- Mejora de la información que ofrece la imagen para la interpretación humana
- Almacenado, representación para facilitar el entendimiento automático, et. al

Pasamos a ver algunos ejemplos de diferentes formas que tenemos de modificar las propiedades de una imagen.

Colores y tonalidades Existen una gran variedad de técnicas para modificar los colores de una imagen. Una de las mas utilizadas es la correcciónγ. El ojo humano bajo condiciones habituales (ni mucha luz, ni muy poca) percibe la potencia siguiendo una función tipo gamma. Imágenes que no estén codificadas con gamma quiere decir que tenemos demasiados píxeles brillantes o oscuros que el ojo no es capaz de diferenciar, al corregir la imagen podemos distinguir objetos de una forma mucho mas fácil a simple vista.[RoyChoudhury, 2014]

Figura 1.2: Imagen obtenida de Wikipedia [Wikipedia, 7 30] sobre la corrección gamma

Movimiento Estas modificaciones consisten básicamente en movimiento, como inversión o reflexión [SketchUp, 2018].

$$V_{inv}(m,n) = V_I(m,(n_{max}-n)+1)$$

$$V_{refl}(m,n) = V_I((m_{max}-m)+1,n)$$

En este ejemplo V_{inv} sería la imagen invertida y V_{refl} sería la reflexión de la imagen.

Interpolación La interpolación consiste en dado una imagen tratar de estimar valores de intensidad para puntos en los que no conocemos su valor

[Pérez and Valente, 2018]. Podemos seguir diferentes criterios para aproximar este valor:

- Atendiendo al valor del vecino mas cercano.
- Interpolación lineal, que considera los 4 píxeles mas cercanos.
- Interpolación bicúbica, que considera los 16 píxeles mas cercanos

Segmentación La segmentación consiste en descomponer los datos de la imagen en estructuras significativas que son relevantes para tareas específicas [PREIM, 2007]. Básicamente, asignamos índices a los píxeles para poder identificar al objeto al que pertenecen. El reconocimiento de objetos es una tarea de mucha complejidad que suele mostrar mejores resultados al ser realizada por los humanos [PREIM, 2007]. En cambio, delimitar dichos objetos es una tarea que requiere de una gran precisión, tarea que suelen realizar mejor los ordenadores.

La segmentación de puede abordar a través de múltiples enfoques, en este proyecto usaremos un enfoque basado en mapas auto-organizados.

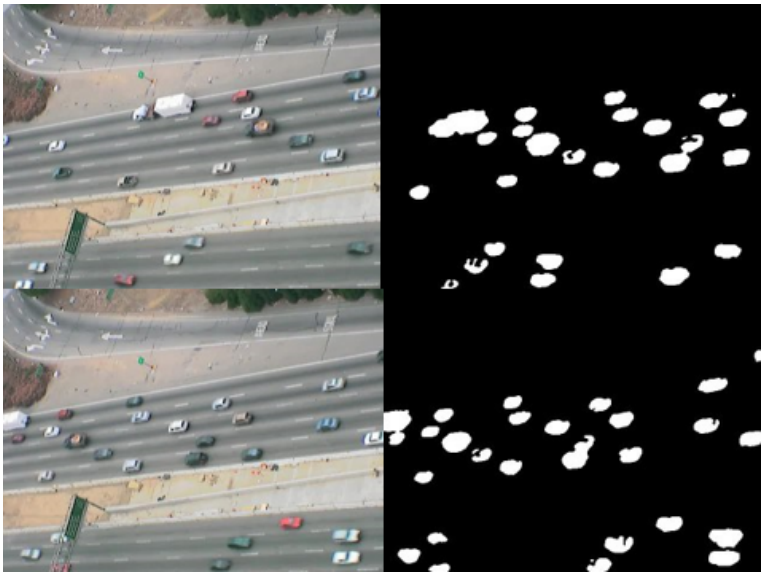


Figura 1.3: Caso real: Segmentación.

1.2. Redes neuronales artificiales

Una redes neuronales artificiales (ANN) es un modelo cuya estructura de capas se inspira en la estructura interconectada de las neuronas del cerebro (capas de nodos conectados). Una ANN puede aprender de los datos para que reconozca patrones, clasifique datos e incluso para que pronostique eventos futuros. [mat, 2019]

Resulta sencillo entender, a grandes rasgos, cómo funciona una ANN. Un modelo de red neuronal podemos verlo como una gran función compuesta [com, 2019], donde cada uno de los nodos definirá una función (como vemos en la Figura 1.3), que en conjunto nos permitirá reconocer patrones no lineales con gran precisión.

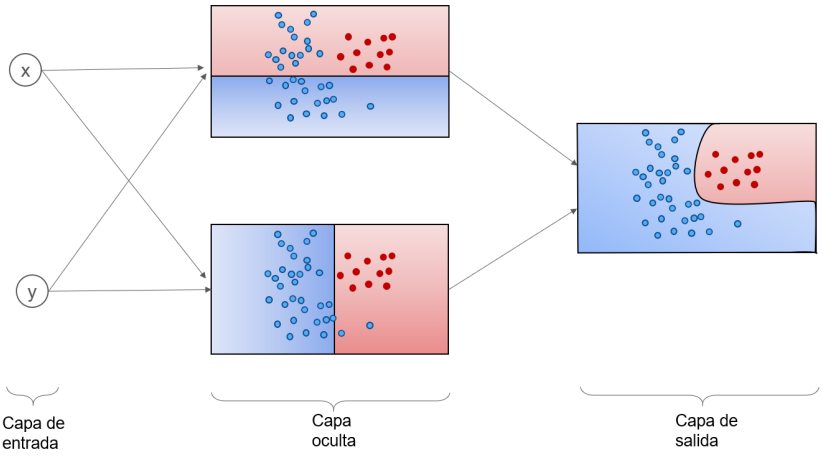


Figura 1.4: Descripción de red neuronal artificial.

1.2.1. Mapas auto-organizados

Los mapas auto-organizados (SOM) se diferencian de las redes neuronales convencionales por utilizar una función de vecindad. La función de vecindad permite preservar las propiedades topológicas de los datos de entrada [Ralhan, 2018]. Los SOM son útiles para poder visualizar datos de múltiple dimensión en baja dimensión, en concreto, nosotros lo usaremos para el proceso de segmentación.

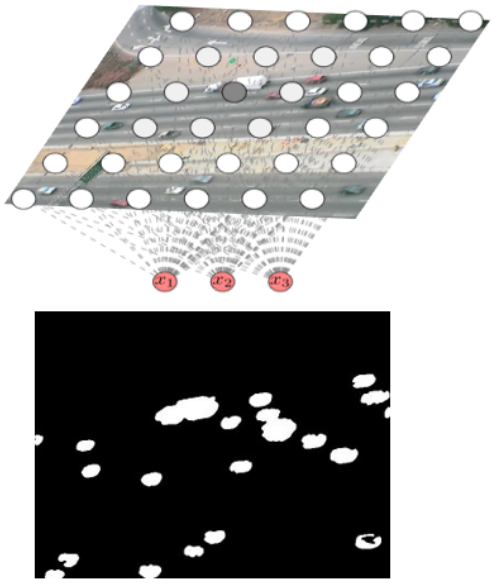


Figura 1.5: Imagen extraída de Wikipedia [MartinThoma, 2016]: Ilustración de SOM.

Los SOM utilizan un aprendizaje competitivo que sigue los siguientes pasos [Kohonen and Honkela, 2007]:

- Crea un mapa de neuronas de forma aleatoria.
- Recibe datos de entrada
  - Por cada neurona del mapa:
    - **Calculamos la distancia** entre el mapa y los datos de entrada.
    - **Guardamos** nuestra **BMU** (*Best matching unit*), que es la neurona que arroja la mínima distancia entre los datos de entrada y el mapa.
  - **Actualiza pesos:** La función que se muestra a continuación es la que se utiliza para actualizar los pesos de una red neuronal.

$$W_v(s+1) = W_v(s) + \Theta(u, v, s)\alpha(s)(D(t) - W_v)$$

donde s es el índice del paso, t es el índice dentro del conjunto entranante, u es el índice de BMU para $D(t)$,α(s) es el coeficiente monótonamente decreciente de aprendizaje y $D(t)$ es el vector de entrada; se asume que v visite todas las neuronas para cada valor de s y t.

- Aumentar s y volver al paso 2, mientras $s < \lambda$.

1.2.2. Aprendizaje profundo y redes convolucionales

Las redes neuronales profundas (aprendizaje profundo) no son mas que ANN pero con muchas capas ocultas. Dentro del campo de la visión artificial es muy común encontrar arquitecturas de redes neuronales profundas, destacan las redes convolucionales [Moreno, 2019].

Las redes convolucionales, a diferencia de una ANN, aplica convoluciones. Las convoluciones nos permiten analizar una imagen por trozos extrayendo los objetos de mayor importancia, esto es mucho menos costoso que conectar todos los píxeles de una imagen con las neuronas de una capa inicial del modelo, donde el número de conexiones entre neuronas se dispararía [Bagnato, 2018].

Uno de los conjuntos de datos generados está formateado para poder entrenar directamente una red llamada Faster R-CNN, que es la red convolucional usada para detectar objetos en imágenes que destaca por su balance de rendimiento y precisión [Shaoqing Ren, 2016].

1.3. Algoritmo de Kalman

El algoritmo de Kalman se utiliza en el campo de la visión por ordenador para el seguimiento de objetos. El algoritmo de Kalman es una excelente herramienta matemática que no solo permite estimar sino filtrar de manera óptima una señal. Este algoritmo está constituido por una etapa de predicción seguida de una etapa de corrección aplicada al proceso de detección de objetos.
[José Ancizar Castañeda Cárdenas, 2013].

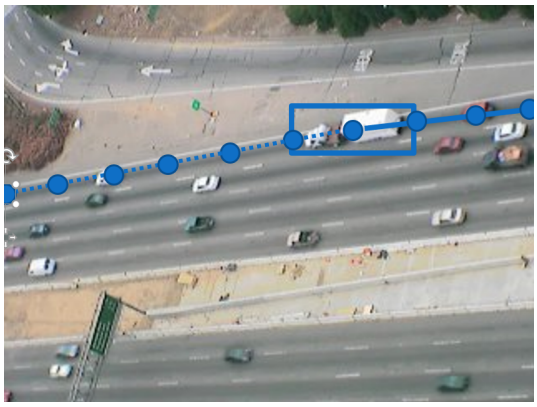


Figura 1.6: Ilustración del algoritmo de Kalman para una trayectoria.

1.4. Principios metodológicos

Los principios metodológicos seguidos para estructurar, planificar y controlar el desarrollo del proyecto han sido los siguientes:

El método científico es el método principal en el que se basa en el trabajo; esta consitutido por dos pilares principales: reproducibilidad y refutabilidad. Si nuestro experimento sigue el método científico, deberá ser verificable, seguir un razonamiento riguroso y hacer observaciones empíricas. Es un método necesario para conseguir teorías estables mediante el conocimiento [Tamayo, 2004].

Las etapas que integran el método científico son: Definición y planteamiento del problema, planteamiento de hipótesis, Contraste de hipótesis, y finalmente, comunicar el resultado alcanzado por la investigación [Castán, 2014].

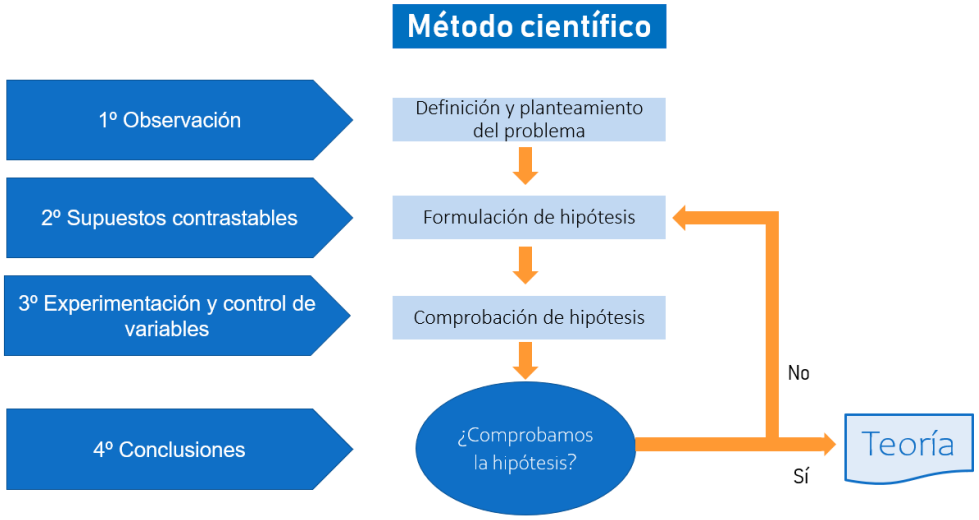


Figura 1.7: Ilustración del método científico.

Metodología iterativa incremental: El trabajo parte de una base sobre la que iremos añadiendo modificaciones iterativamente, evitando que afecten al trabajo ya realizado, para ajustar el proyecto a los requisitos que hemos fijado. Para cada incremento tendremos 4 fases: Análisis, Desarrollo, Instauración y Validación del resultado. [Caycho, 2019]

Metodología de implementación Durante la implementación se han seguido buenas prácticas de programación como: Diseñar una batería de pruebas para testar en la aplicación al realizar mejoras, modularizar, código comprensible, sencillo, modificable, extensible, documentado et al.
[Manuel Guillermo García Sandoval1 and Fuentes4, 2019]

Parte III

Desarrollo

Capítulo 2

Desarrollo y análisis del programa

Contenido

2.1 Detección de objetos 31

2.2 Extracción de trayectorias 33

2.3 Desarrollo interfaz 36

2.3.1 Paso 1: Definir etiquetas 37

2.3.2 Paso 2: Etiquetado de trayectorias 39

2.3.3 Paso 3: Métricas y almacenamiento de extracción 46

Sinopsis

En este apartado acometemos el desarrollo y principal objetivo del trabajo. Explicaremos en detalle las diferentes partes de la herramienta desarrollada. Empezaremos hablado del proceso seguido para la detección de los objetos en las imágenes. A continuación, veremos como se realiza la asociación de los objetos detectados a sus respectivas trayectorias. Seguidamente veremos cómo se ha desarrollo de la interfaz. Finalmente revisaremos los resultados que ofrece esta herramienta. Concluiremos

este capítulo hablando sobre los resultados obtenidos al entrenar una red Faster R-CNN usando los diferentes conjuntos de datos generados.

En la figura 1.1 se observa un diagrama con la visión global del programa.

2.1. Detección de objetos

El método que vamos a utilizar para la identificación de posibles objetos dentro de la imagen es una segmentación basada en mapas auto-organizados, como habíamos comentado anteriormente. Esta segmentación, apodada FSOM, ha sido desarrollada por el grupo de investigación donde trabajan mis tutores de proyecto [López-Rubio et al., 2011].

La diferencia de esta aproximación con un clasificador clásico radica en que en vez de buscar un determinado objeto en la imagen para marcarlo y etiquetarlo, simplemente vamos a buscar posibles objetos dentro de la imagen.

FSOM consiste de 5 pasos fundamentalmente:

Inicialización Creamos e inicializamos el modelo.

Corregir función vecindad Vamos a utilizar una función de vecindad Gausiana que se ajustará para cada instante de tiempo en función del "neighbourhood radius"$\delta(n)$ y de la distancia topológica de cada píxel con la BMU.

Segmentación Utilizamos el mapa auto-organizado para identificar si un píxel pertenece o no a un posible objeto.

Reducir ruido Este apartado se puede dividir en dos partes. En primer lugar haremos una umbralización a las probabilidades de los píxeles de la imagen, convertimos en 0 todas aquellas probabilidades menores de 0.5 a 0 (píxeles que no pertenece a ningún objeto) y en 1 (píxeles que sí pertenece a ningún objeto) las mayores de 0.5. El segundo lugar eliminaremos los objetos demasiado pequeños como para poder ser objetos, y rellenaremos los objetos que puedan tener 0 en su interior.

Extracción de resultados Finalmente delimitaremos el área mínima rectangular que permite englobar a los diferentes objetos.

En la figura 2.1 se muestra, mediante un diagrama de bloques, las fases que sigue la segmentación basada en mapas auto-organizados.

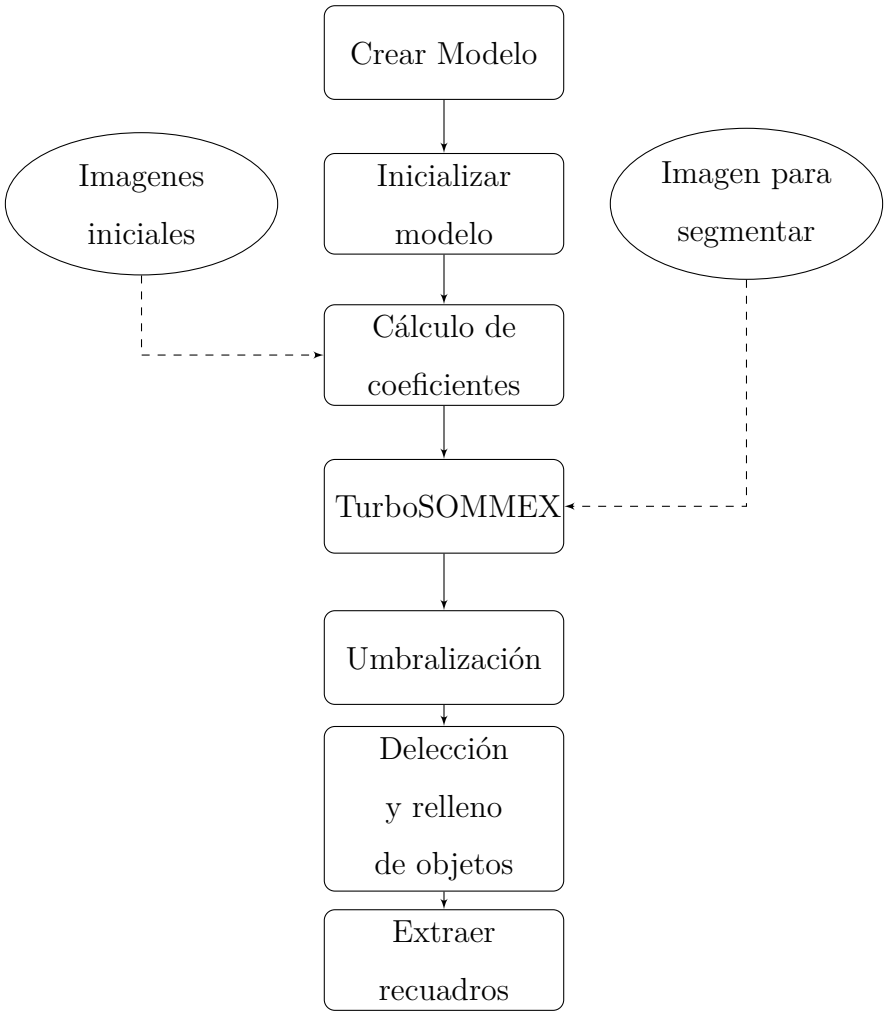


Figura 2.1: Segmentación FSOM

A continuación vemos un ejemplo seleccionado para ver los resultados obtenidos durante la segmentación.

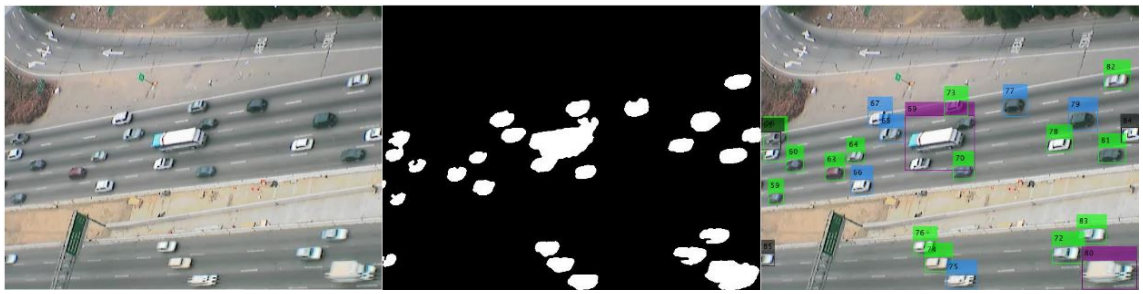


Figura 2.2: Ejemplo segmentación y extracción de recuadros

En concreto, esta muestra es el fotograma 132 del vídeo que hemos utilizado para testar la aplicación [administration, 2019]. Se ha seleccionado este fotograma porque muestra un problema asociado a la detección de objetos, el solapamiento.

Un solapamiento cosiste en la fusión de varios objetos cercados, en estos casos el modelo nos identifica un único objeto, en vez de tres. Encontramos solapamientos en dos casos para este vídeo: en adelantamientos y con la aparición de objetos grandes, como el autobús.

2.2. Extracción de trayectorias

Una vez detectados los objetos en las imágenes pasaremos a unir las apariciones de un mismo objeto con su trayectoria. Para ello nos serviremos del algoritmo de Kalman. En nuestra implementación del algoritmo de Kalman, empezaremos detectando los diferentes objetos que aparecen en la imagen (mediante la segmentación comentada anteriormente). Seguidamente, haremos una predicción en base a las diferentes trayectorias que tenemos, teniendo los resultados de las predicciones de Kalman y las posiciones reales"de los objetos encontrados diferenciamos dos escenarios [VISION, 2019]:

Encontramos un objeto Predecimos el estado actual del objeto y vemos que hemos detectado un objeto (mediante la segmentación) a muy poca distancia

de esa predicción. Consideraremos el objeto segmentado como el estado real que nos servirá para corregir posibles desviaciones de esta trayectoria. En este caso además guardaremos la información de esta detección.

No detectamos un objeto Cuando no detectamos el objeto, el algoritmo de Kalman solo se basa en el estado anterior del objeto para predecir su ubicación actual.

Teniendo el siguiente estado del objeto de las trayectorias anteriores solo falta comprobar nuevas trayectorias y eliminar trayectorias finalizadas. Añadiremos nuevas trayectorias para los objetos a los que no se les asocie ninguna trayectoria, y eliminaremos aquellos objetos que lleven demasiado tiempo sin aparecer, o que las predicciones de Kalman indiquen que se han salido de los límites del vídeo. En la figura 2.3 vemos del proceso seguido para traquear las trayectorias.

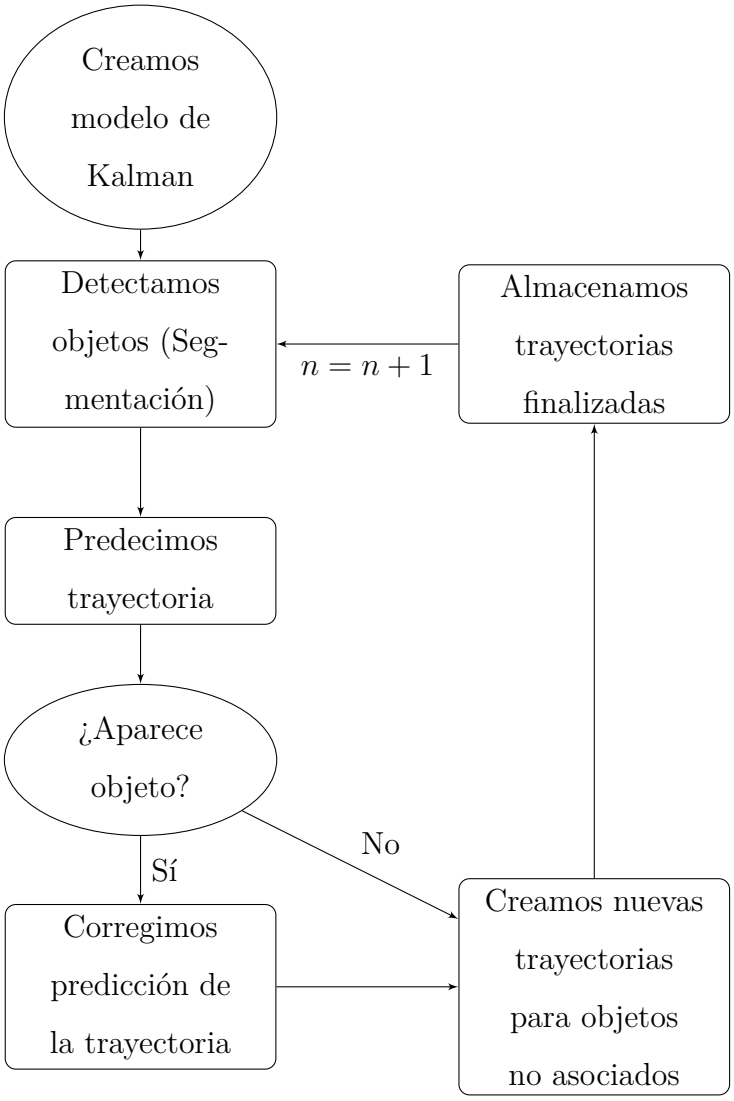


Figura 2.3: Diagrama algoritmo de Kalman

Notamos que el algoritmo de Kalman no da buenos resultados cuando aparecen escenas de atascos. El algoritmo de Kalman no es capaz de seguir la trayectoria de los objetos cuando se detienen durante un periodo de tiempo. Esta será una limitación a tener en cuenta a la hora de usar esta herramienta.

2.3. Desarrollo interfaz

Definimos los siguientes requisitos para poder hacer que la experiencia del usuario, con esta herramienta, fuera fácil y sencilla.

Definir clases Poder definir, borrar y editar clases

Selección múltiple Seleccionar múltiples trayectorias de una misma etiqueta

Editar recuadros Ofrecer una opción sencilla a la hora de modificar un recuadro que no ha sido detectado correctamente.

Visuaización Constituye una parte fundamental de la herramienta que el usuario sea capaz de saber con un simple vistazo:

- Las trayectorias identificadas que no han sido seleccionadas
- Las trayectorias identificadas que han sido seleccionadas
- Las trayectorias identificadas que han sido seleccionadas y etiquetadas

En un principio decidimos que este programa podría ser fácil de utilizar usando simplemente la terminal. Al ejecutar el código introducías las clases que ibas a utilizar y aparecía una ventana con los fotogramas. En esta ventana podías hacer diferentes combinaciones de clics de ratón para acceder a las funcionalidades, por ejemplo:

Clic izquierdo Seleccionar objeto

Clic derecho Otra opción

- **Clic izquierdo** Cambiar tamaño del recuadro y seleccionar objeto
- **Clic derecho** otra opción
  - **Clic izquierdo** Etiquetar objetos seleccionados

- **Clic derecho** Siguiente fotograma

El problema que tenía este diseño es muy evidente, se vuelve mas difícil de usar conforme se añaden más funcionalidades. Por este motivo decidimos desarrollar interfaz.

Parte del código existente tuvo que ser modificado para poder integrarse dentro de la interfaz. Se decidió utilizar Matlab appdesigner por las facilidades que suponía para el desarrollo, ya se había desarrollado gran parte del código en Matlab, nos permitiría cumplir con los tiempos que habíamos designado al inicio del proyecto. Además de las funcionalidades esperadas para crear una interfaz encontramos mas que nos permitirían hasta crear un instalador para nuestro programa.

La herramienta está constituida de 3 pasos, cada pestaña de la interfaz esa ligada a un paso: Definición de etiquetas, etiquetado de trayectorias y extracción de información. En caso de tener dudas sobre la utilización de la interfaz, se puede hacer uso de los botones de ayuda. Se han ubicado botones en las diferentes pasos de la aplicación, la ayuda será específica del paso en el que nos encontremos; también podemos acceder a la ayuda general que se encuentra en la pestaña de ayuda.

2.3.1. Paso 1: Definir etiquetas

El proceso de definir las etiquetas es el primer paso de la interfaz. Podemos ver un ejemplo de cómo se muestra este paso en la interfaz en la figura 2.5. En este paso el usuario hará 3 acciones:

- Seleccionar vídeo
- Seleccionar carpeta donde queremos volcar los resultados
- Definir etiquetas

Tanto para elegir el vídeo, como el directorio de salida, tendremos que hacer clic en el botón **Find o Choose**. Al hacer clic, se despliega una ayuda en la que podremos navegar para elegir la carpeta de salida y el vídeo que queremos. Una vez seleccionados solo tenderemos que darle al botón de aceptar y nos aparecerá en la interfaz la ruta que hemos seleccionado.

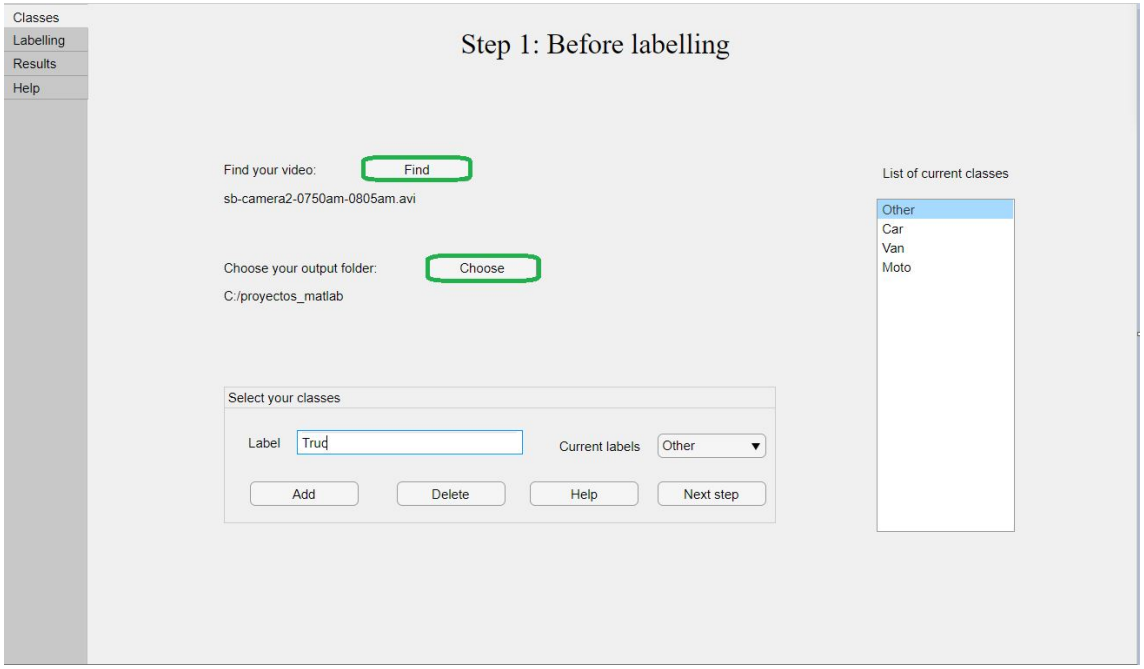


Figura 2.4: Elegir carpetas

El proceso de añadir etiquetas básicamente consiste de dos opciones, añadir etiqueta y eliminar etiqueta. Para añadir una etiqueta debemos escribir el nombre de la etiqueta en el campo de texto *label* y hacer clic en el botón **add**. Podemos comprobar que esta etiqueta se ha añadido correctamente porque ahora nos aparecerá en el recuadro *list of current classes*, y también debe de aparecer al desplegar *Current labels*. Se ha añadido la opción para que el usuario pueda añadir clases durante el etiquetado de objetos. La interfaz no permite introducir dos clases que se llamen igual.

El recuadro de *list of current classes* es meramente informativo, nos permite visualizar muy rápidamente las clases que tenemos, en cambio, el desplegable *current labels* lo utilizaremos para eliminar clases no deseadas. Para eliminar una clase seleccionaremos la clase no deseada en el desplegable *current labels* y haremos clic en **delete**.

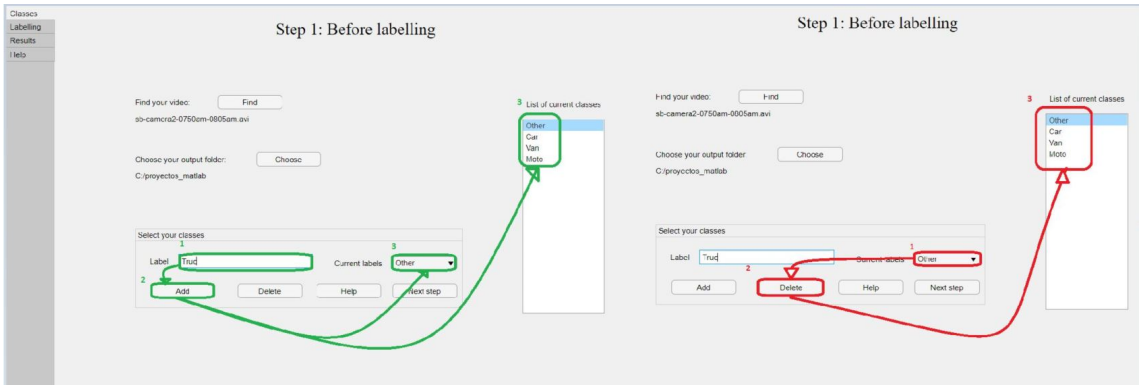


Figura 2.5: Añadir etiqueta y borrar etiqueta

En caso de tener problema con la interfaz podemos recurrir al botón de ayuda **help**, que nos ofrecerá información específica del paso en el que nos encontramos. Antes de poder pasar al siguiente paso debemos haber: seleccionado vídeo, seleccionado directorio y añadido al menos una clase, de lo contrario nos aparecerá un mensaje notificando que falta información si intentamos avanzar.

Para pasar al siguiente paso haremos clic en el botón **Next step**, y si hemos rellenado toda la información, nos aparecerá una ventana indicando que se está inicializando los modelos.

2.3.2. Paso 2: Etiquetado de trayectorias

En esta segunda pestaña de nuestra interfaz.

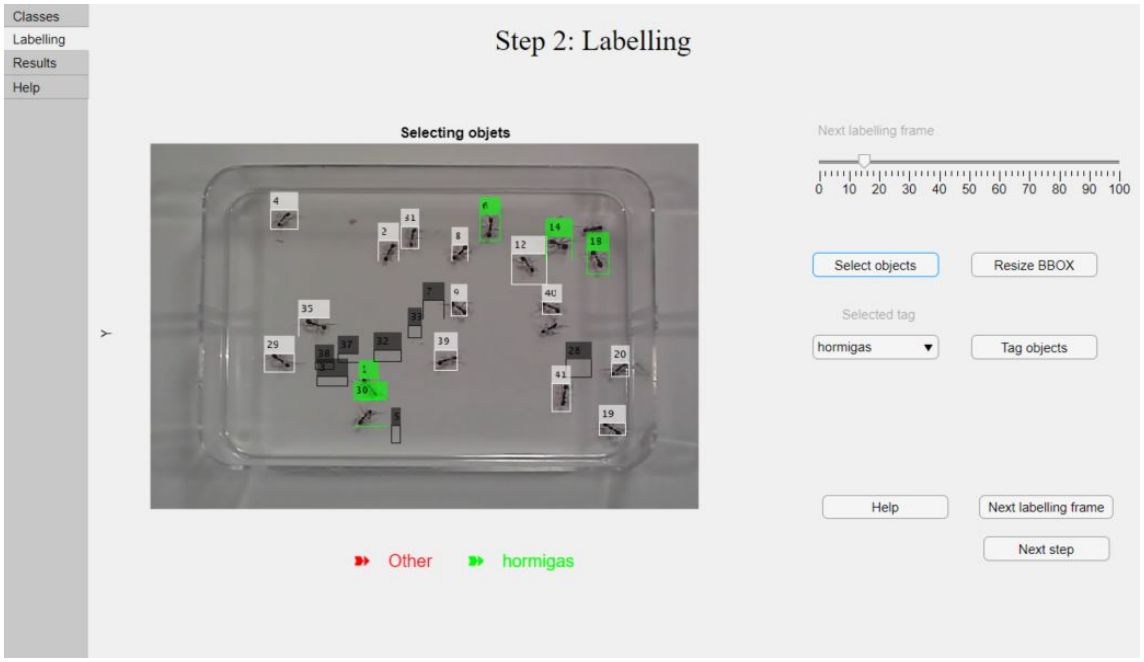


Figura 2.6: Ejemplo del paso 2 de la interfaz

A la izquierda un gran recuadro donde se mostrarán las imágenes del vídeo que estamos estudiando, y a la derecha, las diferentes funcionalidades que se han desarrollado para el etiquetado, como podemos ver en la figura figura 2.6. Podemos notar que las clases definidas anteriormente nos aparecen coloreadas bajo la imagen, estos colores se corresponden con los que usaremos para identificar las clases ya etiquetadas.

El etiquetado de las trayectorias es un proceso constituido por dos etapas. En primer lugar, seleccionaremos los objetos en la imagen una vez parado el video en un frame, y a continuación los etiquetaremos. En la imagen 2.7 podemos ver en los puntos uno y dos la apariencia de la interfaz cuándo el video está en movimiento y cuando no respectivamente.

Para seleccionar los objetos en la imagen, podemos usar tanto el botón **Select objets**, que nos permitirá seleccionar múltiples objetos, como el botón **Resize**

BBOX, que nos permitirá cambiar el tamaño de recuadro de un único objeto y seleccionarlo. En cualquiera de los dos casos nos aparecerá una ventana emergente como la que se muestra en la imagen 2.7, en el punto 3.

Cambiar tamaño del recuadro y selección Para cambiar el tamaño de un recuadro tenemos que pulsar donde queremos tener una futura esquina de nuestro recuadro y arrastrar el ratón hasta la esquina contraria. De esta forma el recuadro mas cercano será sustituido por el nuevo que hemos definido.

Selección múltiple de objetos Para seleccionar múltiples objetos simplemente debemos pulsar con el botón izquierdo del ratón la parte mas próxima al centro del recuadro de los objetos que deseemos etiquetar.

Una vez seleccionado un objeto, la interfaz cambia el color del recuadro de ese objeto de color negro a blanco; de esta forma podemos intuir que el objeto ya ha sido correctamente seleccionado.

Figura 2.7: Etiquetado: Paso 1 (Puntos suspensivos indican que son acciones sobre ventanas emergentes.)

Una vez acabemos la selección de los objetos (tienen que ser del mismo tipo), elegiremos en el desplegable **Selected tag** la etiqueta para los objetos seleccionados. Para etiquetar los objetos, con la etiqueta seleccionada, haremos clic en el botón **Tag objects**, como se muestra en la imagen 2.12.

Figura 2.8: Etiquetado: Paso 2

Tras etiquetar un fotograma podemos pasar al siguiente, para ello solo tenemos que hacer clic en en el botón **Next labelling frame**. El número de fotogramas que avanzaremos en el vídeo lo podremos modificar haciendo uso de la barra **Next labelling frame**. En caso de equivocarnos etiquetando un objeto solo tendremos que volver a seleccionarlo y asignarle la etiqueta correctamente.

Se ha comprobado el correcto funcionamiento del sistema con diferentes, como por ejemplo:

- Detección de renacuajos

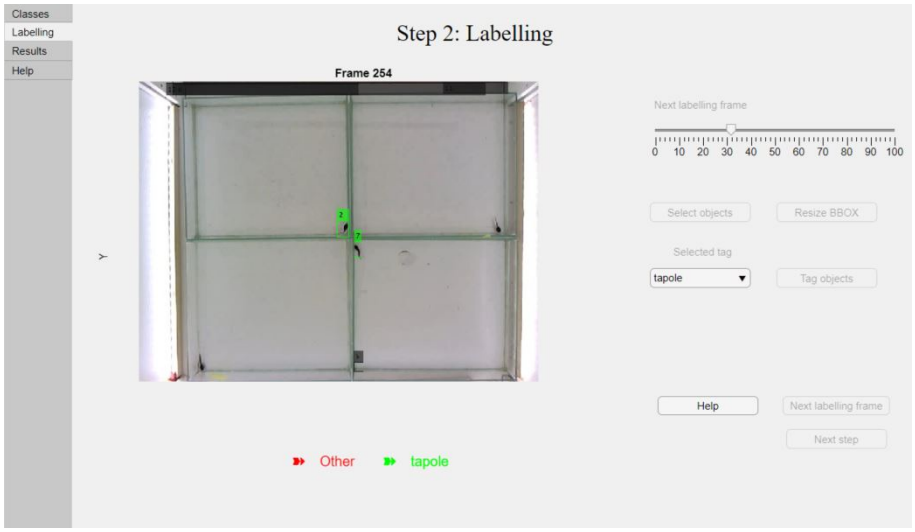


Figura 2.9: Detección de trayectorias en renacuajos

- Detección del movimiento de pez cebra

Figura 2.10: Detección de trayectorias en pez cebra

- Detección de trayectorias de hormigas

Figura 2.11: Detección de trayectorias en hormigas

- Detección de diferentes tipos de vehículos

Figura 2.12: Detección de trayectorias en vehículos

Paso 3.1: Extracción de datos

Entre el paso 2 y el paso 3 realizaremos la extracción de datos, que se inicia haciendo clic en el botón **Next step** una vez finalizado todo el etiquetado. Este proceso es costoso computacionalmente debido a la estructura del objeto que tenemos que recorrer, y por toda la información que guardamos. Se ha monitorizado el tiempo que tarda en generar todos los conjuntos de datos para 500 fotogramas y 240 trayectorias (10020 objetos detectados), tarda un total de 3 minutos 15 segundos.

Generaremos tres conjuntos de datos:

- Automáticos
- Semi-automáticos
- Manuales

En los datos manuales solo extraeremos la aparición del objeto cuando fue etiquetado (extraemos un objeto por cada vez que etiquetamos), en los datos semi-automáticos extraemos las trayectorias de aquellos objetos que hemos etiquetado (extraemos una trayectoria por cada vez que etiquetamos), y finalmente para los datos automáticos extraeremos todos los datos de todas las trayectorias estén o no etiquetados (extraemos todas las trayectorias, las no etiquetadas se guardarán como *Other*).

2.3.3. Paso 3: Métricas y almacenamiento de extracción

Esta última vista de nuestra interfaz esta dedicada a mostrar algunos datos sobre los conjuntos generados. Por un lado, mostramos los conjuntos de datos brutos generados(kModels), cada uno tiene un máximo de 100 trayectorias almacenadas. Vemos el número de objetos que hemos etiquetado para cada clase y para cada

objeto. Y se ofrece tanto un acceso directo a los conjuntos de imágenes recortadas, como la opción de reiniciar el proceso.

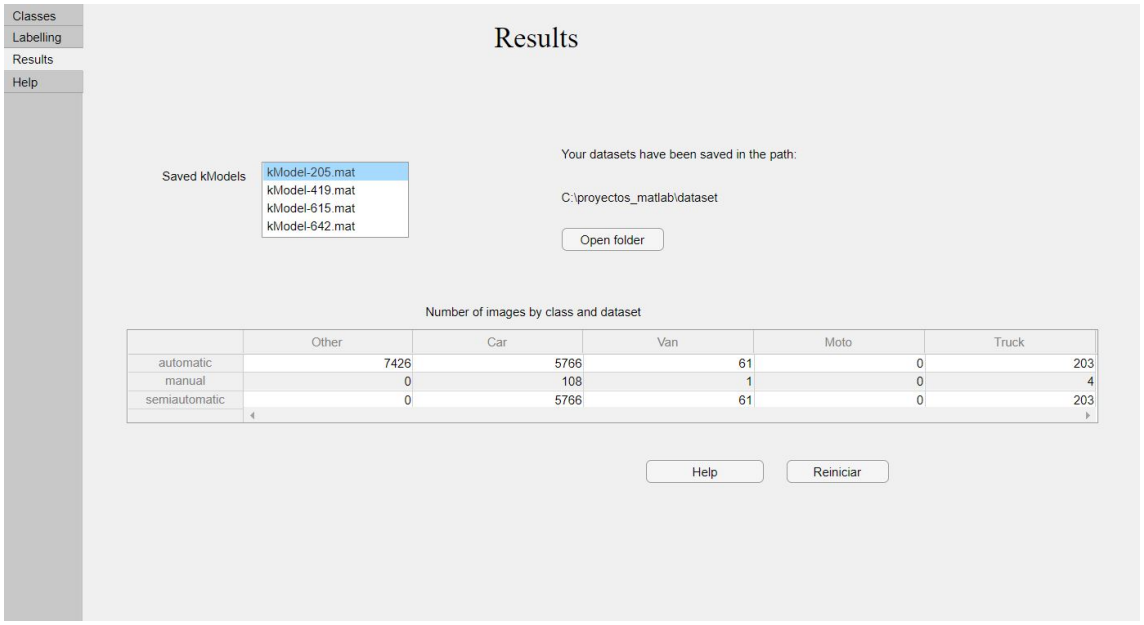


Figura 2.13: Ejemplo del paso 3 de la interfaz

Una vez llegado a este punto podemos acceder a la carpeta que habíamos seleccionado al principio como directorio de trabajo y comprobar que se ha generado la siguiente estructura de carpetas 2.14.

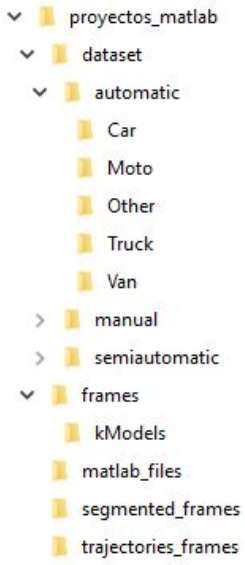


Figura 2.14: Estructura de carpetas generada.

En este caso la carpeta que habíamos elegido es **proyectos_matlab**. Dentro de esta carpeta se han generado 5 carpetas:

datasets En esta carpeta se almacenarán las imágenes recortadas para los 3 conjuntos de datos (automatic, semiatuomatic y manual). Dentro de las carpetas de los conjuntos de datos, se ordenan en función de su etiqueta las imágenes recortadas.

frames Dentro de esta carpeta almacenamos los fotogramas del vídeo sin procesar, además, guardaremos dentro de la carpeta *kModels*, las trayectorias detectadas en bruto.

matlab_files Dentro de esta carpeta guardamos 3 archivos **.mat**. Cada archivo corresponde a uno de de los conjuntos de datos generados (automatic, semiatuomatic y manual), y están diseñados para entrenar directamente un modelo **Faster R-CNN**.

segmented_frames En esta carpeta guardaremos los resultados de la segmenta-

ción que arroja el FSOM.

trajectories_frames En esta carpeta guardaremos los fotogramas junto con todos los recuadros detectados.

Conclusiones y líneas futuras

En este proyecto, se ha tratado de aplicar en conjunto un método de segmentación basado en mapas auto-organizados, y el filtro de Kalman para detectar trayectorias de objetos, sobre el campo del etiquetado de datos, en el dominio de la visión por ordenador, una de las principales ramas de la inteligencia artificial.

El objetivo principal del proyecto era detectar las trayectorias de los objetos, objetos que después etiquetaríamos manualmente, permitiendo obtener datos etiquetado fiables de forma rápida. De esta forma podríamos generar conjuntos de datos etiquetados de forma fiable y rápida. Como objetivos secundarios se establecieron que la herramienta debería de ser fácil de utilizar y que permita comprobar rápidamente la validez de los datos.

La interfaz resultante muestra el potencial que supone detectar las trayectorias de los objetos, permitiéndonos etiquetar gran cantidad de datos en poco tiempo, lo que se traduce en una reducción de los costes del etiquetado de datos. Esta herramienta supone una alternativa muy interesante a las herramientas comentadas en el estado del arte. La herramienta dispone de las funcionalidades mas básicas cumplen con el objetivo de ser fáciles de utilizar para el usuario. Y parte de los datos han sido generados especialmente en un formato para que se puedan entrenar directamente un modelo Faster R-CNN lo que nos permite conocer la calidad de los datos fácilmente.

Pese a que los objetivos del proyecto se han cumplido conforme a lo esperado, durante el desarrollo se han detectado posibles mejoras que podemos ver como **futuras**

líneas de investigación.

Mejora de la fiabilidad En lo referente a la fiabilidad de los datos, encontramos varias líneas de investigación. Una posible línea de investigación, consistiría en evolucionar la detección de objetos para que el resultado final no fueran meramente recuadros, que pudiésemos etiquetar figuras mas complejas. Otra futura línea de investigación puede estar centrada en resolver el problema que suponen los solapamientos. Para el caso concreto del vídeo utilizado [administration, 2019], podríamos tratar de solventar el problema de los solapamientos mediante un algoritmo de adelgazamiento, tras la segmentación y la reducción de ruido.

Al realizar un adelgazamiento obtenemos un resultado como el que se muestra para la figura 2.15, donde los objetos solapados corresponden con las ramas del grafo que tienen mas de 3 ramas. No parece difícil poder diseñar algún algoritmo capaz de detectar cuando hay una rama con mas de 3 terminaciones, y que divida esa rama por la mitad, diferenciando los diferentes objetos, estableciendo como rama principal de grafo la última que se ha conseguido esqueletizar.

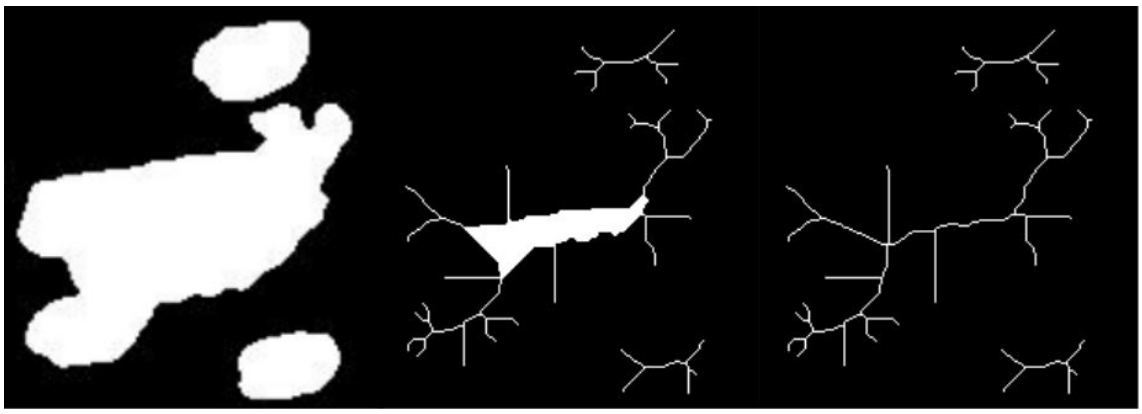


Figura 2.15: Ejemplo algoritmo de adelgazamiento.

Como vemos este algoritmo de adelgazamiento no nos hace perder infor-

mación del tamaño original detectado gracias a las diferentes ramas que nos ayudarán a marcar las esquinas de los objetos detectados.

Mejora de tiempos En lo referente a optimización del tiempo, encontramos posibles líneas de investigación en: la mejora de los algoritmos para recorrer la estructura donde tenemos almacenadas las trayectorias, y aumento en la velocidad del vídeo durante el etiquetado.

Mejoras de la interfaz En lo referente a la interfaz podemos hacer algunas mejoras, como por ejemplo, cambiar la barra que usamos para seleccionar el número de fotogramas que queremos avanzar, añadir funcionalidades como selección múltiple, que al definir un recuadro seleccione todos aquellos objetos que se encuentren en su interior, o añadir un paso mas en el que se entrene automáticamente un modelo Faster R-CNN que nos permita conocer las métricas para los conjuntos de datos generados.

Jose Rodríguez Maldonado

21 de noviembre de 2019

Bibliografía

[def, 2005] (2005). *Fundamentos de la imagen digital*. CELA-UNAM. http://ignorantisimo.free.fr/PDI/docs/PDI2005_APT_02A-Capitulo_II_(PARTE_1)_Fundamentos_de_la_imagen_digital.pdf.

[fun, 2005] (2005). *Fundamentos de la imagen digital*. CELA-UNAM. http://ignorantisimo.free.fr/PDI/docs/PDI2005_APT_02A-Capitulo_II_(PARTE_1)_Fundamentos_de_la_imagen_digital.pdf.

[com, 2019] (2019). tdx.

[mat, 2019] (2019). *¿Qué es una red neuronal?* MATLAB. https://es.mathworks.com/discovery/neural-network.html.

[Abhishek Dutta and Zisserman, 9 01] Abhishek Dutta, A. G. and Zisserman, A. (2019-09-01). *VGG Image Annotator (VIA)*. University of Oxford. http://www.robots.ox.ac.uk/~vgg/software/via/.

[Academy, 2019] Academy, K. (2019). *El método científico*. AMGEN FUNDATION. https://es.khanacademy.org/science/biology/intro-to-biology/science-of-biology/a/the-science-of-biology.

[administration, 2019] administration, F. H. (2019). *Next Generation Simulation (NGSIM)*. FHWA. https://ops.fhwa.dot.gov/trafficanalysistools/ngsim.htm.

[Adrián Peña-Peñate*, 2016] Adrián Peña-Peñate*, Luis Guillermo Silva Rojas, R. A. N. (2016). *Módulo de filtrado y segmentación de imágenes médicas digitales para el proyecto Vismedic*. Revista Cubana de Ciencias Informáticas. http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2227-18992016000100002.

[AltexSoft, 2018] AltexSoft (05-2018). *How to Organize Data Labeling for Machine Learning: Approaches and Tools*. kdnuggets. https://www.kdnuggets.com/2018/05/data-labeling-machine-learning.html.

[Andra Petrovai and Nedevschi, 9 01] Andra Petrovai, A. D. C. and Nedevschi, S. (2019-09-01). *Semi-Automatic Image Annotation of Street Scenes*. https://cv.utcluj.ro/tl_files/cv/publications/Semi-Automatic%20Image%20Annotation%20of%20Street%20Scenes.pdf.

[AWS, 2019] AWS (2019). *Using AI to Make Fully Autonomous Trucks*. AWS machine learning. https://aws.amazon.com/es/machine-learning/customers/innovators/tusimple/.

[Axa, 2019] Axa (2019). *Fast Image Data Annotation Tool (FIAT)*. Axa. https://github.com/christopher5106/FastAnnotationTool.

[Bagnato, 2018] Bagnato, J. I. (29-11-2018). *¿Cómo funcionan las Convolutional Neural Networks? Visión por Ordenador*. https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/.

[Brownlee, 2019] Brownlee, J. (2019). *A Gentle Introduction to the ImageNet Challenge (ILSVRC)*. Machine Learning Mastering. https://machinelearningmastery.com/.

[Castán, 2014] Castán, Y. (2014). *INTRODUCCIÓN AL MÉTODO CIENTÍFICO Y SUS ETAPAS*. Instituto Aragonés de Ciencias de la Salud. http://www.ics-aragon.com/cursos/salud-publica/2014/pdf/M2T00.pdf.

[Caycho, 2019] Caycho, E. A. (12 abril, 2019). *Ciclos de vida iterativo e incremental, ¿Qué son?* cdpSchool. https://www.cursodireccionproyectos.com/2018/07/ciclos-de-vida-iterativo-e-incremental-que-son/.

[Demush, 2019] Demush, R. (02-09-2019). *A Brief History of Computer Vision and convolutional neural networks*. hackernoon. https://hackernoon.com/.

[Drummond, 2017] Drummond, C. (2017). *El impacto económico mundial de la Inteligencia Artificial*. TICBeat. https://www.ticbeat.com/tecnologias/el-impacto-economico-mundial-de-la-inteligencia-artificial-infografia/.

[Floren, 2012] Floren, C. (23-07-2012). *The Advantages and Disadvantages of Using Crowdsourcing to Improve Your Ecommerce Business*. myecommerce. http://myecommerce.biz/blog/2012/07/.

[Fukushima, 1982] Fukushima, K. (1982). *Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position*. NHK Broadcasting Science Research Laboratories, Kinuta, Setagaya, Tokyo, Japan. https://www.rctn.org/bruno/public/papers/Fukushima1980.pdf.

[Heras, 2018] Heras, J. M. (21-12-2018). *Fases del Proceso de Machine Learning*. IArtificial. https://iartificial.net/fases-del-proceso-de-machine-learning/.

[HUBEL, 1981] HUBEL, D. H. (08-12-1981). *EVOLUTION OF IDEAS ON THE PRIMARY VISUAL CORTEX, 1955-1978: A BIASED HISTORICAL*

ACCOUNT. Harvard Medical School, Department of Neurobiology, Boston, Massachusetts, U.S.A. https://www.nobelprize.org/uploads/2018/06/hubel-lecture.pdf.

[Hubel and Wiesel, 1959] Hubel, D. H. and Wiesel, T. N. (Oct 1959). *Receptive fields of single neurones in the cat's striate cortex*. Journal of physiology. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1363130/.

[ImageNet, 2012] ImageNet (2012). *ImageNet Challenge 2012 Analysis*. ImageNet. http://image-net.org/challenges/LSVRC/2012/analysis/.

[INFAIMON, 1 18] INFAIMON (2018-01-18). *Visión por computador: qué es y cuáles son sus usos más comunes*. INFAIMON. https://blog.infaimon.com/vision-computador-soluciones-permite/.

[INGEDATA, 2019a] INGEDATA (19-03-2019a). *Annotating Data: Crowdsourcing Platforms vs. Outsourcing Companies*. INGEDATA. https://www.ingedata.net/blog/annotating-data-crowdsourcing-vs-outsourcing.

[INGEDATA, 2019b] INGEDATA (19-03-2019b). *The Cost of AI: Why Labelling Data In-House Is a Bad Move*. INGEDATA. https://www.ingedata.net/blog/the-cost-of-ai-why-labelling-data-in-house-is-a-bad-move.

[José Ancizar Castañeda Cárdenas, 2013] José Ancizar Castañeda Cárdenas, Manuel Antonio Nieto Arias, V. A. O. B. (2013). *ANÁLISIS Y APLICACIÓN DEL FILTRO DE KALMAN A UNA SEÑAL CON RUIDO ALEATORIO*. Ingeniería electrónica, Universidad Tecnológica de Pereira, Pereira, Colombias. https://revistas.utp.edu.co/index.php/revistaciencia/article/view/8241.

[Juan de Antonio, 2018] Juan de Antonio, fundador y CEO de Maxi Mobility, h. d. f. c. C. e. E. (2018). *Machine learning y auge*

de Inteligencia Artificial. FORBES. https://www.forbes.com.mx/machine-learning-y-auge-de-inteligencia-artificial/.

[Kohonen and Honkela, 2007] Kohonen, T. and Honkela, T. (2007). *Kohonen netowork*. Scholarpedia. http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2227-18992016000100002.

[Labelbox, 9 01] Labelbox, I. (2019-09-01). *Labelbox*. Labelbox, Inc. https://labelbox.com/.

[LeCun, 1989] LeCun, Y. (12-09-1989). *Backpropagation applied to handwritten zip code recognition*. ATandT Bell Laboratories Holmdel, NJ, 07733 USA. http://yann.lecun.com/exdb/publis/pdf/lecun-89e.pdf.

[Lewis, 2013] Lewis, S. (02-01-2013). *The first scanned image*. Shot by Scot. http://www.scottlewisphotography.eu/?p=1788.

[López-Rubio et al., 2011] López-Rubio, E., Luque-Baena, R., and Domínguez, E. (2011). *Foreground detection in video sequences with probabilistic self-organizing maps*.

[Lowe, 1999] Lowe, D. G. (January 1999). *Object Recognition from Local Scale-Invariant Features*. University of British Columbia. https://www.cs.ubc.ca/~lowe/papers/iccv99.pdf.

[Manuel Guillermo García Sandoval1 and Fuentes4, 2019] Manuel Guillermo García Sandoval1, Héctor David Ariza Torrado2, M. L. P. and Fuentes4, A. S. F. (2019). *BUENAS PRÁCTICAS APLICADAS A LA IMPLEMENTACION COLABORATIVO DE APLICATIVOS WEB*. Revista mundo FESC, Universidad de Pamplona. https://dialnet.unirioja.es/descarga/articulo/5351802.pdf.

[MartinThoma, 2016] MartinThoma (2016). *Self-organizing-map*. https://commons.wikimedia.org/wiki/File:Self-organizing-map.svg.

[MATLAB, 2019] MATLAB (2019). *Comparación de MATLAB y R en cuanto al prototipado y la implementación de análisis*. MATLAB. https://es.mathworks.com/discovery/matlab-vs-r.html.

[Moreno, 2019] Moreno, C. G. (2019). *¿Qué es el Deep Learning y para qué sirve?* indra. https://www.indracompany.com/es/blogneo/deep-learning-sirve.

[Naimat, 2016] Naimat, A. (August 2016). *La Inteligencia Artificial impulsará el PIB mundial un 14 % en 2030 por sus efectos en la productividad y en el consumo*. O'Reilly Media, Inc. https://www.oreilly.com/library/view/the-new-artificial/9781492048978/.

[Nakhuda, 2019] Nakhuda, D. (09-01-2019). *5 Reasons You Shouldn't Label Training Data In-House*. MightyAI. https://mighty.ai/blog/5-reasons-you-shouldnt-label-training-data-in-house/.

[opencv, 9 01] opencv (2019-09-01). *Computer Vision Annotation Tool (CVAT)*. opencv. https://github.com/opencv/cvat.

[Osuna, 2019] Osuna, R. (2019). *Fundamentos de la imagen digital*. CELA-UNAM. https://www2.uned.es/personal/rosuna/resources/photography/ImageQuality/fundamentos.imagen.digital.pdf.

[PASCAL2, 2019] PASCAL2 (2019). *The PASCAL Visual Object Classes Homepage*. EU-funded PASCAL2 Network of Excellence. http://host.robots.ox.ac.uk/pascal/VOC/.

[PREIM, 2007] PREIM, BERNHARD Y BARTZ, D. (2007). *Visualization in Medicine: Theory, Algorithms, and Applications*. https:

//cv.utcluj.ro/tl_files/cv/publications/Semi-Automatic%20Image%20Annotation%20of%20Street%20Scenes.pdf.

[Pérez and Valente, 2018] Pérez, P. and Valente, M. (2018). *Curso de introducción al procesamiento de imágenes radiológicas en ámbito clínico*. https://www.famaf.unc.edu.ar/~pperez1/manuales/cim/index.html.

[PwC, 2017] PwC (2017). *La Inteligencia Artificial impulsará el PIB mundial un 14 % en 2030 por sus efectos en la productividad y en el consumo*. PricewaterhouseCoopers International Limited (PwCIL). https://www.pwc.es/es/sala-prensa/notas-prensa/2017/la-inteligencia-artificial-impulsara-pib-mundial.html.

[RAINERI, 2019] RAINERI, S. (28-02-2019). *Top Outsourcing Disadvantages*. The balance. https://www.thebalancesmb.com/top-outsourcing-disadvantages-2533777.

[Ralhan, 2018] Ralhan, A. (18-02-2018). *Self Organizing Maps*. https://towardsdatascience.com/self-organizing-maps-ff5853a118d4.

[Roberts, 1963] Roberts, L. (January 1963). *Machine Perception of Three-Dimensional Solids*. researchgate. https://www.researchgate.net/publication/220695992_Machine_Perception_of_Three-Dimensional_Solids.

[RoyChoudhury, 2014] RoyChoudhury, A. K. (2014). *Using instruments to quantify colour*. Principles of Colour and Appearance Measurement. https://www.sciencedirect.com/science/article/pii/B9780857092298500073.

[Shaoqing Ren, 2016] Shaoqing Ren, Kaiming He, R. G. J. S. (6-1-2016). *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. https://arxiv.org/abs/1506.01497.

[SketchUp, 2018] SketchUp (2018). *Flipping, Mirroring, Rotating and Arrays.* Trimble Inc. https://help.sketchup.com/en/sketchup/flipping-mirroring-rotating-and-arrays.

[Soria, 2019] Soria, P. E. (2019). *INTRODUCCIÓN AL PROCESADO DIGITAL DE SEÑALES. Conversión A/D, D/A*. E.T.S.E Universitat de València. https://www.uv.es/soriae/tema_1_pds.pdf.

[Surma, 2019] Surma, G. (02-2019). *Image Generator - Drawing Cartoons with Generative Adversarial Networks.* Towards Data Science. https://towardsdatascience.com/image-generator-drawing-cartoons-with-generative-adversarial-networks-45e814ca9b6

[Tamayo, 2004] Tamayo, M. (2004). *El proceso de la investigación científica.*

[Viola and Jones, 2001] Viola, P. and Jones, M. (2001). *Rapid Object Detection using a Boosted Cascade of Simple Features.* Mitsubishi Electric Research Labs and Compaq CRL. https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf.

[virajmavani, 2019] virajmavani (01-09-2019). *Anno-Mage: A Semi Automatic Image Annotation Tool.* virajmavani. https://virajmavani.github.io/saiat/.

[VISION, 2019] VISION, M. (2019). *Using Kalman Filter for Object Tracking.* MATLAB. https://es.mathworks.com/help/vision/examples/using-kalman-filter-for-object-tracking.html.

[Wikipedia, 7 30] Wikipedia (2019-07-30). *Corrección gamma.* https://es.wikipedia.org/wiki/Correcci%C3%B3n_gamma.

[wkentaro, 9 01] wkentaro (2019-09-01). *Image Polygonal Annotation with Python.* wkentaro. https://github.com/wkentaro/labelme.

[Xin-Jing Wang and Ma, 2006] Xin-Jing Wang, Lei Zhang, F. J. and Ma, W.-Y. (2006). *AnnoSearch: Image Auto-Annotation by Search*. Microsoft Research Asia, 49 Zhichun Road, Beijing (100080), China. https://www.researchgate.net/publication/4246334_AnnoSearch_Image_Auto-Annotation_by_Search.